The present work was submitted to the LuFG Theory of Hybrid Systems

Bachelor of Science Thesis

Subtropical Satisfiability for Polynomial Constraint Sets

Giang Lai

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. Christina Büsing
Additional Advisor:
Jasper Nalbach, M. Sc.

Aachen, 28.3.2022

Abstract

We elaborate an idea for efficient determination of satisfiability of quantifier-free non-linear real arithmetic formulas which we also refer to as (multivariate) polynomial constraint sets. The main goal is transforming a set of polynomial constraints into a single equisatisfiable polynomial equation which is suitable for the subtropical real root finding method [FOSV17] [Stu15]. We analyze a proposal for such a transformation and prove that it is incorrect for conjunctions of equational constraints which are a crucial part of SMT solving. We present alternative ideas and perspectives as well, but a solution or proof for the impossibility of the main idea has yet to be found. We still lay out the necessary details for subtropical satisfiability checking from the *International Symposium on Frontiers of Combining Systems* [FOSV17] and depict an optional method for constructing a model using the cylindrical algebraic decomposition [Jir95]. We test implementation ideas for utilities and efficiency improvements for the SMT-RAT solver that are a byproduct of this research and discussions. Finally, we also conduct other benchmarks for the edge case of a single polynomial equation to see how the subtropical real root finding method [Stu15] compares to another used strategy of SMT-RAT. The results coincide with the foundings in [FOSV17] and [Stu15]; they show that SMT-RAT using the subtropical real root finding method is significantly more efficient for this case. The empirical evidence implies that an efficient realization of the main idea could lead to great improvement in SMT solving efficiency.

Erklärung

Hiermit versichere ich, dass ich die vorgelegte Arbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Giang Lai
Aachen, den 28. März 2022

Acknowledgements

This thesis about possibly elevating SMT solving would not have been possible without the support and ideas of my supervisor and examiner Prof. Dr. Erika Ábrahám, Dr. Thomas Sturm, and Dr. Hamid Rahkooy. Independently researching and working on SMT-RAT have not only given me a new experience that helped me to improve my learning skills and knowledge, but they also have shown me what kind of mistakes can occur while being on your own and how to avoid these in the future. I want to hereby thank them very much for proposing this topic, providing the necessary literature, and being open to discussions. I especially want to emphasize my appreciation to Prof. Dr. Erika Ábrahám for giving me this opportunity and being an excellent supervisor. I also want to thank Prof. Dr. Christina Büsing for agreeing to become my second examiner for this thesis. I would like to give my thanks to Jasper Nalbach for the introduction and access to SMT-RAT and for providing helpful material. Last but not least, I want to thank my family and friends for their encouraging support which has been proven as greatly helpful especially during the times of the corona pandemic.

Contents

1 Introduction **9**
1.1 Notation 11

2 Preliminaries **13**
2.1 Subtropical Satisfiability 13
2.2 Cylindrical Algrebraic Decomposition 23

3 The Transformation and Equisatisfiability **33**
3.1 Transformation Properties 34
3.2 Further Analysis 37
3.3 Model Construction for a Polynomial Equation 39
3.4 Outlook 40

4 Utility for SMT-RAT **43**
4.1 Tests 43
4.2 Other Implementations 45

5 Conclusion **47**

Bibliography **49**

Chapter 1

Introduction

Satisfiability modulo theory solvers, or SMT solvers for short, have a variety of uses as they are capable of tackling any kind of problem that can be described as a first-order formula. It ranges from pure mathematical questions, like finding an intersection of two circles, to more practical real-world applications, like program analysis or software verification [Fit12]. Those applications have important roles because they ensure the correctness of the intended behavior of computer programs. This does not only include programs of businesses or industries with economic interests but also securing a stable infrastructure for the safety of a city or treating people with the use of medical devices. To name a few examples, the barrier Maeslantkering [maeb] is handled by a computer system that evaluates weather data and measures the sea level. It is responsible for opening the barrier, making port traffic possible, and for closing the barrier, protecting South Holland with a population of over 3.7 million people (as of January 2021) [Sta] against disastrous storm surges [maea]. The next example is a case which occurred during 1985 to 1987 [LT93]. The Therac-25, produced in 1982, was a medical machine that was used for radiation therapy. It killed cancer cells by applying doses of radiation directly to the patient. Due to a race condition, i.e. a programming error, the Therac-25 applied hundred times greater radiation doses than necessary which resulted into severe injuries or even fatal cases.

Example 1.0.1. Description of the intersection of two circles

$$\varphi(x,y) := x^2 + y^2 - 1 = 0 \wedge (x+2)^2 + y^2 - 5 = 0$$

This formula is satisfied by the interpretation $x = 0$ and $y = 1$.

An SMT instance can be understood as an advanced SAT instance which allows for the interpretation of functions, predicates, and quantification over variables using any kind of domain. It has therefore a greater expressiveness as an SAT instance. SMT solving is NP-hard because SAT solving is NP-complete already (see Cook's theorem) proven by Stephen A. Cook in 1971 [Coo71]. On top of that, depending on the theory, the task may become undecidable. This is to be expected because satisfiability of first-order formulas, in general, is already undecidable [Grä11].

We consider quantifier-free non-linear real arithmetic (QF NRA in short) which is decidable proven by Alfred Tarski [Tar98]. In other words, we look at formulas (i.e. statements) without quantified variables which are described using polynomials over the reals. We refer to them as polynomial constraints and a finite collection of QF NRA formulas as polynomial constraint sets. A polynomial constraint set is said to be satisfiable if the conjunction of its elements is satisfiable. Notice that disjunctions and conjunctions within QF NRA formulas are equivalent to satisfiability of polynomial constraint sets: For example, for two atomic QF NRA formulas φ, ψ is the satisfiability of $\varphi \vee \psi$ equivalent to the satisfiability of $\{\varphi\}$ or $\{\psi\}$ and the satisfiability of $\varphi \wedge \psi$ is equivalent to $\{\varphi\} \cup \{\psi\}$.

In the mid-chapter, we discuss and elaborate a certain idea for the improvement of SMT solving. It proposes to compress a set of polynomial constraints into a single equisatisfiable polynomial equational constraint for the application of the subtropical real root finding method, described in [FOSV17] [Stu15]. We explain the necessary parts of the subtropical satisfiability from [FOSV17] and how to construct a model for a satisfiable polynomial equational constraint using methods of the cylindrical algebraic decomposition for univariate polynomials. At the end of this paper, we depict some statistics of other related implementations within the SMT solver SMT-RAT, which is under the care of the research group *Theory of Hybrid Systems at RWTH Aachen University*, and conclude our results.

1.1 Notation

We use the notation from *International Symposium on Frontiers of Combining Systems* [FOSV17] within this paper for coherence:

- f,g,h are polynomials.
- x,y,z are real-valued variables.
- $f(x), g(x)$ refer to polynomial functions.
- a represents a real value.
- $\boldsymbol{x}$ is a vector $(x_1,...,x_d)$ of variables $x_1,...,x_d$ (notice the **bold** notation).
- $\boldsymbol{p}, \boldsymbol{q}$ are points such as $(p_1,...,p_d) \in \mathbb{R}^d$.
- For $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^d$, $\overline{\boldsymbol{pq}}$ is the set $\{((1-\lambda)\boldsymbol{p} + \lambda\boldsymbol{q}) \in \mathbb{R}^d : \lambda \in [0,1]\}$
- For a value $a \in \mathbb{R}$ and point $\boldsymbol{p} \in \mathbb{R}^d$, $a^{\boldsymbol{p}}$ is the point $(a^{p_1},...,a^{p_d})$.
  If $\boldsymbol{p} = (1,...,1)$ then we abbreviate $a^{\boldsymbol{p}}$ to $\boldsymbol{a} = (a,...,a) \in \mathbb{R}^d$
  (for example: $\boldsymbol{0} = (0,...,0) \in \mathbb{R}^d$).
- For a vector $\boldsymbol{x}$ and a point $\boldsymbol{p} \in \mathbb{R}^d$, $\boldsymbol{x}^{\boldsymbol{p}}$ is the vector $(x_1^{p_1},...,x_d^{p_d})$.
- $f_{\boldsymbol{p}} \in \mathbb{Z} \setminus \{0\}$ is the coefficient at the monomial (of a polynomial f) that has exactly the powers $\boldsymbol{x}^{\boldsymbol{p}}$. Implicitly, this means that we have a strict variable ordering to prevent ambiguity. A monomial can also be referred as $f_{\boldsymbol{p}} \cdot \boldsymbol{x}^{\boldsymbol{p}}$.

We also introduce additional notation:

- F represents a set of polynomials.
- $\boldsymbol{r} \in \mathbb{R}^d$ is a root of a polynomial f.
- $\xi \in \mathbb{R}$ represents a real value.
- Substitutions of a variable x in a polynomial f for a real value ξ are written as $f[x/\xi]$.
- φ, ψ denote polynomial constraints.
- Φ denotes a polynomial constraint set.
- Substitutions of variables $\boldsymbol{x} = x_1,...,x_d$ in a logical formula φ for values $M(\boldsymbol{x})$, where M is a structure that maps $x_1,...,x_d$ to real values, is written as $\varphi[\boldsymbol{x}/M(\boldsymbol{x})]$.
- $\lambda \in [0,1]$ is a real-valued variable between zero and one.
- For two points $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^d$ the univariate polynomial $\lambda_i \in \mathbb{Q}[\lambda]$ is equal to:

$$(1-\lambda)p_i + \lambda q_i$$

Chapter 2

Preliminaries

2.1 Subtropical Satisfiability

The Broad Idea

This section introduces the *subtropical satisfiability* proposed in [FOSV17] in a summarized manner along with different examples and additional explanations.

As the name suggests, *subtropical real root finding* is a method to calculate a solution to a (polynomial) equation. This method is incomplete but still very efficient in its use and is therefore optimal as a heuristic [FOSV17]. The idea for finding a solution to a polynomial equation $f = 0$, where $f \in \mathbb{Q}[x_1,...,x_d]$, can be broken down into three steps [FOSV17]:

1. Compute the value $f(\mathbf{1})$. Evaluate $f(\mathbf{1})$ and follow the substeps accordingly:
   - If $f(\mathbf{1}) = 0$ then we have found a solution.
   - If $f(\mathbf{1}) > 0$ then start over but consider $-f$ instead.
   - Otherwise, proceed with the next step.
2. Find a point $\boldsymbol{p}$ with **positive entries** such that $f(\boldsymbol{p}) > 0$.
3. Find a point $\boldsymbol{r} \in \overline{\mathbf{1}\boldsymbol{p}}$ such that $f(\boldsymbol{r}) = 0$.

As we gather a negative and a positive value of the polynomial function, we can make use of the fact that polynomial functions are continuous and therefore the intermediate value theorem guarantees a root, i.e. a solution to the equation.
The second step is done by performing a reduction from the original problem to a linear programming problem (also called LP). An LP aims to maximize a function $\boldsymbol{c}^T \cdot \boldsymbol{x}$ for $\boldsymbol{c} \in \mathbb{R}^d$; the solution $\boldsymbol{x}$ is restricted by linear constraints [RAUa]. We only need to solve the LP to some relatively little extent since an optimal solution is not required. The solution to the reduced problem can then be used to construct a solution for the original equation.
The reason why a point with **positive entries** is required in the second step is not arbitrary, nor a restriction due to the computation. There are practical reasons: In chemistry and biology are constraints with very large polynomials common and the desired solution needs to be positive for any practical use [Stu15]. Of course, there is

a way to find more general solutions, i.e. non-positive solutions. However, explaining this in more detail later would exceed the scope of this thesis. For the interested reader, we leave a reference to [FOSV17] instead.

There are three additional uses that can be extracted from the above mentioned idea structure:

- If we are only interested in satisfiability of inequations and finding a respective model then we can omit step 1 and step 3.
- If we are only interested in satisfiability of inequations then we only need to determine satisfiability of the LP.
- If we are only interested in satisfiability of equations then we can omit step 3.

A modular implementation of the steps of the subtropical real root finding method makes it possible to input constraints not only of the form $f = 0$. It is also possible to input constraints of the form $f \sim 0$ where $\sim \in \{>, \geq, <, \leq, \neq\}$, using the following adjustments:

- $f > 0$ does not need to be adjusted.
- The weak constraint $f \geq 0$ becomes the strict constraint $f > 0$. If no solution is found due to the incompleteness of this method then the remaining constraint $f = 0$ cannot be applied in the first place as this also requires a positive point of f therefore the answer of $f \geq 0$ would become "Unknown".
- The constraint $f < 0$ is changed to $-f > 0$.
- The case for the constraint $f \leq 0$ can be inferred from the above mentioned cases.
- The last constraint $f \neq 0$ becomes $f > 0 \vee -f > 0$.

The Intuition for Finding Positive Values of a Polynomial

Any non-constant polynomial f with a single indeterminate x has this property:

$$\lim_{x \to \infty} f(x) = \begin{cases} \infty, & \text{if leading coefficient is greater than 0} \\ -\infty, & \text{if leading coefficient is less than 0} \end{cases}$$

The term with the leading coefficient includes by definition the greatest exponent and grows therefore faster than the sum of the remaining terms. In other words, there exists some value x_0 such that for all $x > x_0$ the absolute value of the leading term at x is greater than the sum of all other terms at x. Now, because $x > x_0 > 0$ the leading coefficient is the only deciding factor whether the function is positive or negative at $f(x)$. The following examples provide more insight on how we can utilize this knowledge to find positive values.

Example 2.1.1.

$$f(x,y,z) := x^3 - 3y^2 - 60z$$

This polynomial function has more than one indeterminate, but by choosing two variables and setting them to 0 we receive a polynomial function with a single indeterminate:

- $f(0{,}0{,}z) = -60z$, the leading coefficient is less than 0.
- $f(0{,}y{,}0) = -3y^2$, the leading coefficient is less than 0.
- $f(x{,}0{,}0) = x^3$, the leading coefficient is greater than 0.

However, polynomial functions in multiple indeterminates are not that conveniently constructed. In general cases, variables appear together in the majority of terms.

Example 2.1.2.

$$f(x{,}y) := -x^4 + xy^2 - y$$

Setting one of the variables to 0 results in a polynomial with a negative leading coefficient, as the only monomial with a positive coefficient vanishes. Letting both variables approach infinity at the same rate causes the monomial $-x^4$ to dominate. The solution is letting one variable grow faster than the other. Suppose we substitute y for y^2. The substitution results in the polynomial $-x^4 + xy^4 - y^2$. Letting both variables approach infinity at the same rate in that polynomial causes the monomial xy^4 to dominate with the desired positive coefficient. A positive point can then be found by finding a large enough base a and calculating $f(a{,}a^2)$ [FOSV17]:

$$a = 2 \implies f(2{,}4) = -16 + 32 - 4 = 12 > 0$$

Example 2.1.3.

$$f(x{,}y) := -x^3y - x^2y^3 + xy$$

At first, one might assume that no positive point can be found using only positive values for x and y. Also substituting x or y with greater powers does not work, as the other monomials also grow faster.
The solution here is to substitute x and y with x^{-1} and y^{-1} respectively instead. By doing so, each monomial becomes their multiplicative inverse, i.e. $-\frac{1}{x^3y}, -\frac{1}{x^2y^3}$, and $\frac{1}{xy}$ respectively. As the quotients with greater exponents grow faster, the fractions become smaller, resulting in $\frac{1}{xy}$ becoming the dominant term. Of course, the expression that results from this substitution is not a polynomial because exponents must be either zero or a natural number. Effectively, we do not actually substitute anything but rather use this observation to find the correct input. By finding a large enough base a, a positive value of f can be computed:

$$a = 2 \implies f(a^{-1}, a^{-1}) = f(\frac{1}{2}, \frac{1}{2}) = -\frac{1}{16} - \frac{1}{32} + \frac{1}{4} = \frac{5}{32} > 0$$

An algorithmic implementation of these intuitions for finding positive points is the reduction to linear programming, i.e. the subtropical method.

Finding Positive Values of a Polynomial

Before explaining how the reduction works, we recall some concepts from [FOSV17].

Definition 2.1.1. Frames

The *frame* of a polynomial $f = \sum f_{\boldsymbol{p}} \cdot \boldsymbol{x}^{\boldsymbol{p}} \in \mathbb{Q}[x_1,...,x_d]$ is defined as:

$$\text{frame}(f) = \{\boldsymbol{p} \in \mathbb{N}_0^d : f_{\boldsymbol{p}} \neq 0\}$$

We can create a partition of this set by changing the condition $f_{\boldsymbol{p}} \neq 0$ to $f_{\boldsymbol{p}} > 0$ and $f_{\boldsymbol{p}} < 0$, i.e. the sign of the coefficient is considered as well:

$$\text{frame}^+(f) = \{\boldsymbol{p} \in \text{frame}(f) : f_{\boldsymbol{p}} > 0\} \quad \text{frame}^-(f) = \{\boldsymbol{p} \in \text{frame}(f) : f_{\boldsymbol{p}} < 0\}$$

Consider the following polynomial:

$$f_{ex} := -x^5y^3 + 2x^4y + 3x^3y^2 + x^2y - 4y^3 - 5$$

The frames of f_{ex} are:

- $\text{frame}(f_{ex}) = \{(5,3), (4,1), (3,2), (2,1), (0,3), (0,0)\}$
- $\text{frame}^+(f_{ex}) = \{(4,1), (3,2), (2,1)\}$
- $\text{frame}^-(f_{ex}) = \{(5,3), (0,3), (0,0)\}$

Definition 2.1.2. Convex Hull

A *convex* set $M \subseteq \mathbb{R}^d$ fulfills the following condition:

$$\forall \boldsymbol{p}, \boldsymbol{q} \in M : \overline{\boldsymbol{p}\boldsymbol{q}} \subseteq M$$

Let $S \subseteq \mathbb{R}^d$ be a set of points. The *convex hull* of S is the smallest convex superset of S or equivalently:

$$\text{conv}(S) = \bigcap_{S \subseteq M \subseteq \mathbb{R}^d} M \qquad \text{where } M \text{ is convex}$$

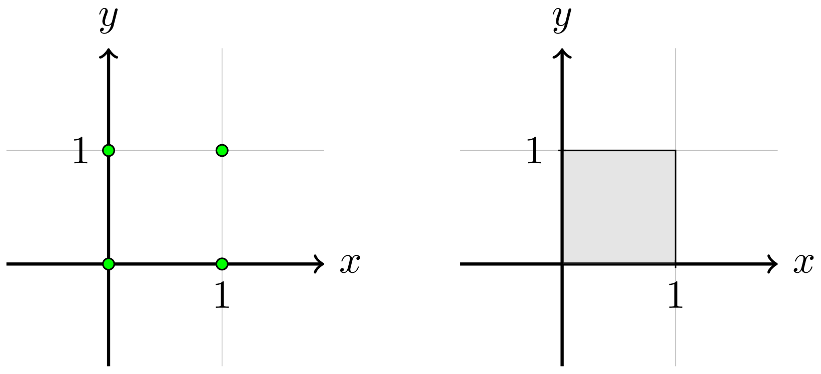

Figure 2.1: $\{(0,0), (0,1), (1,0), (1,1)\}$ (left) and its convex hull $\{(x,y) : x,y \in [0,1]\}$

Definition 2.1.3. Newton Polytope

The *Newton polytope* of a polynomial f is defined as:

$$\operatorname{newton}(f) = \operatorname{conv}(\operatorname{frame}(f))$$

Figure 2.2 shows the Newton polytope of f_{ex} as a shaded area. The green dots refer to the elements of $\operatorname{frame}^{+}(f_{ex})$ and the black dots to the elements of $\operatorname{frame}^{-}(f_{ex})$.

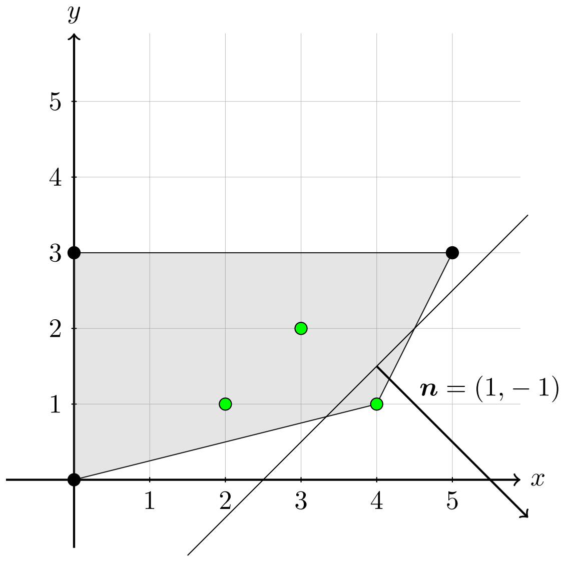

Figure 2.2: Newton polytope of f_{ex} and a hyperplane with direction $(1, -1)$

A polytope is a geometrical object and can be understood as a generalization of polygons or polyhedrons in terms of dimensions. T. Sturm has proven in [Stu15] that in order to find a positive value, a hyperplane $h : \boldsymbol{n}^T\boldsymbol{x} + c$ such that $\boldsymbol{n}^T\boldsymbol{p} > \boldsymbol{n}^T\boldsymbol{q}$ where $\boldsymbol{p} \in \operatorname{frame}^{+}(f)$ and $\boldsymbol{q} \in \operatorname{frame}(f) \setminus \{\boldsymbol{p}\}$ is required. In other words, we need to compute a hyperplane that separates an element of the positive frame of a polynomial from the rest of its Newton polytope. The normal vector component of such a hyperplane contains the information to construct a solution which lets the desired monomial to dominate over the others. The following lemma is taken from [FOSV17] Lemma 2 which builds on [Stu15] Lemma 4.

Lemma 2.1.1. Let f be a polynomial, $\boldsymbol{p} \in \operatorname{frame}(f)$, and $\boldsymbol{n}$ a normal vector such that $\forall \boldsymbol{q} \in \operatorname{frame}(f) \setminus \{\boldsymbol{p}\} :\ \boldsymbol{n}^T\boldsymbol{p} > \boldsymbol{n}^T\boldsymbol{q}$. Then there exists an $a_0 \in \mathbb{R}^{+}$ such that for all $a \geq a_0$ the following holds:

1. $|f_{\boldsymbol{p}} \cdot a^{\boldsymbol{n}^T\boldsymbol{p}}| > \sum_{\boldsymbol{q} \in \operatorname{frame}(f) \setminus \{\boldsymbol{p}\}} |f_{\boldsymbol{q}} \cdot a^{\boldsymbol{n}^T\boldsymbol{q}}|$ (monomial $f_{\boldsymbol{p}} \cdot \boldsymbol{x}^{\boldsymbol{p}}$ dominates)

2. $\operatorname{sign}(f(a^{\boldsymbol{n}})) = \operatorname{sign}(f_{\boldsymbol{p}})$

Proof. Assume the above mentioned conditions hold.

Let $max = \max\{\boldsymbol{n}^T\boldsymbol{q} : \boldsymbol{p} \neq \boldsymbol{q} \in \text{frame}(f)\}$. Consider the following quotient:

$$\begin{aligned}\frac{|f_{\boldsymbol{p}} \cdot a^{\boldsymbol{n}^T\boldsymbol{p}}|}{\sum_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} |f_{\boldsymbol{q}} \cdot a^{\boldsymbol{n}^T\boldsymbol{q}}|} &\geq \frac{|f_{\boldsymbol{p}}| \cdot a^{\boldsymbol{n}^T\boldsymbol{p}}}{a^{max} \cdot \sum_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} |f_{\boldsymbol{q}}|} \\ &= \frac{|f_{\boldsymbol{p}}|}{\sum_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} |f_{\boldsymbol{q}}|} \cdot a^{\boldsymbol{n}^T\boldsymbol{p}-max}\end{aligned}$$

By our assumptions, we have $\boldsymbol{n}^T\boldsymbol{p} - max > 0$. Analyzing the limit of that expression yields:

$$\lim_{a\to\infty} \frac{|f_{\boldsymbol{p}}|}{\sum_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} |f_{\boldsymbol{q}}|} \cdot a^{\boldsymbol{n}^T\boldsymbol{p}-max} = \frac{|f_{\boldsymbol{p}}|}{\sum_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} |f_{\boldsymbol{q}}|} \cdot \lim_{a\to\infty} a^{\boldsymbol{n}^T\boldsymbol{p}-max} = \infty$$

The limit approaches infinity and therefore the left hand side of the relation in 1. grows faster, i.e. statement 1. is true. The second statement follows from the first one and by the restriction that $a > 0$ therefore $a^{\boldsymbol{n}^T\boldsymbol{p}} > 0$. $\square$

The lemma validates that the problem of finding a positive (or negative) point of a polynomial f can be reduced to finding a hyperplane h that separates a point $\boldsymbol{p} \in \text{frame}^+(f)$ (or $\text{frame}^-(f)$ respectively). This problem can be formulated as an LP instance, without the need of optimization, and might therefore be solved by any LRA (linear real arithmetic) solver. Fontaine and colleagues have shown in [FOSV17] that the corresponding formula is:

$$\varphi(\boldsymbol{p}, \text{frame}(f), \boldsymbol{n}, c) \doteq \boldsymbol{n}^T\boldsymbol{p} + c > 0 \wedge \bigwedge_{\boldsymbol{q}\in\text{frame}(f)\setminus\{\boldsymbol{p}\}} \boldsymbol{n}^T\boldsymbol{q} + c < 0$$

The actual formula, in terms of syntax, generalizes frame(f) with the notion S, but we abuse the notation for the sake of intuition. $\boldsymbol{p}$ and frame(f) are given and therefore the formula has exactly $d+1$ real variables $n_1, n_2, ..., n_d$ and c [FOSV17]. The linear constraint system, which is equivalent to the formula above, is:

$$\begin{aligned} n_1 \cdot p_1 + ... + n_d \cdot p_d + c &> 0 \\ n_1 \cdot q_{1,1} + ... + n_d \cdot q_{1,d} + c &< 0 \\ &\vdots \\ n_1 \cdot q_{m,1} + ... + n_d \cdot q_{m,d} + c &< 0 \end{aligned}$$

where $m = |\text{frame}(f) \setminus \{\boldsymbol{p}\}|$.

As a hyperplane does not necessarily separate exactly one single point, we use a definition to specify those separated sets.

Definition 2.1.4. Face

The *face* of a polytope $P \subseteq \mathbb{R}^d$ in respect to some normal vector $\boldsymbol{n} \in \mathbb{R}^d$ is defined as:

$$\text{face}(\boldsymbol{n}, P) = \{\boldsymbol{p} \in P \,|\, \forall \boldsymbol{q} \in P : \boldsymbol{n}^T\boldsymbol{p} \geq \boldsymbol{n}^T\boldsymbol{q}\}$$

A face with dimension 0 (i.e. the face is a singleton) is referred to as vertex.

Similar statements to the lemma can be applied to faces with higher dimensions. If the sum of the coefficients of the monomials, whose exponent points are elements of that face, is not zero then we can find a value of f with the same sign as the sum by increasing a in $f(a^{\boldsymbol{n}})$.

Corollary 2.1.2. Let f be a polynomial, $\boldsymbol{p}_1, ..., \boldsymbol{p}_m \in \text{frame}(f)$, $\boldsymbol{n}$ a normal vector such that $\forall 1 \leq i \leq m \forall \boldsymbol{q} \in \text{frame}(f) : \ \boldsymbol{n}^T\boldsymbol{p}_i \geq \boldsymbol{n}^T\boldsymbol{q}$, and finally let $\sum_{i=1}^m f_{\boldsymbol{p}_i} \neq 0$. Then there exists an $a_0 \in \mathbb{R}^+$ such that for all $a \geq a_0$ the following holds:

1. $|\sum_{i=1}^m (f_{\boldsymbol{p}_i} \cdot a^{\boldsymbol{n}^T\boldsymbol{p}_i})| > \sum_{\boldsymbol{q} \in \text{frame}(f) \setminus \{\boldsymbol{p}_1, ..., \boldsymbol{p}_m\}} |f_{\boldsymbol{q}} \cdot a^{\boldsymbol{n}^T\boldsymbol{q}}|$
2. $\text{sign}(f(a^{\boldsymbol{n}})) = \text{sign}(\sum_{i=1}^m f_{\boldsymbol{p}_i})$

Proof. For all $\boldsymbol{p}_i, \boldsymbol{p}_j \in \text{frame}(f)$ with $1 \leq i \leq j \leq m$ holds: $\boldsymbol{p}_i \geq \boldsymbol{p}_j \wedge \boldsymbol{p}_i \leq \boldsymbol{p}_j$ and therefore $\boldsymbol{p}_i = \boldsymbol{p}_j$. Since they are all equal, $a^{\boldsymbol{n}^T\boldsymbol{p}_i}$ has the same value for each $\boldsymbol{p}_i$ and we can factor it out:

$$|(\sum_{i=1}^m f_{\boldsymbol{p}_i}) \cdot a^{\boldsymbol{n}^T\boldsymbol{p}}| \qquad \text{where } \boldsymbol{p} \text{ can be any } \boldsymbol{p}_i$$

Using Lemma 2.1.1. it follows the statement because $\sum_{i=1}^m f_{\boldsymbol{p}_i} \neq 0$ is just a constant. The second statement is proven analogously. □

However, if the sum is zero then the monomials $f_{\boldsymbol{p}} \cdot \boldsymbol{x}^{\boldsymbol{p}}$ with $\boldsymbol{p} \in \text{face}(\boldsymbol{n}, \text{newton}(f))$ eliminate each other. Also the sign of $f(a^{\boldsymbol{n}})$ (for increasing a) depends on the other monomials $f_{\boldsymbol{q}} \cdot \boldsymbol{x}^{\boldsymbol{q}}$ where $\boldsymbol{q} \notin \text{face}(\boldsymbol{n}, \text{newton}(f))$.

Full example

Given below is an example to reiterate the rough idea of the procedure of the subtropical method in a conclusive manner.

Example 2.1.4. Determining satisfiability of the constraint $f_{ex} = 0$

Reminder: $f_{ex} = -x^5y^3 + 2x^4y + 3x^3y^2 + x^2y - 4y^3 - 5$

1. Evaluate $f_{ex}(\boldsymbol{1}) = -1 + 2 + 3 + 1 - 4 - 5 = -4$
2. Find a positive point using the subtropical method:
   (a) Iterate through the points of frame$^+(f_{ex})$:
      - $\boldsymbol{p}$ = (2,1): We can see in figure 2.2 that any point that is "inside" the polytope cannot be separated at all.
      - $\boldsymbol{p}$ = (3,2): The same reasoning for the preceding case holds for this one.
      - $\boldsymbol{p}$ = (4,1): We can see in figure 2.2 that the normal vector $\boldsymbol{n} = (1, -1)$ is promising for a separating hyperplane that isolates the vertex (4,1).

   (b) Calculating $\boldsymbol{n}^T\boldsymbol{q}$ for each $\boldsymbol{q} \in \text{frame}(f_{ex})$ yields:
      - $\boldsymbol{q}$ = (5,3) $\rightarrow$ 2
      - $\boldsymbol{q}$ = (4,1) $\rightarrow$ 3

- $\boldsymbol{q} = (3{,}2) \to 1$
- $\boldsymbol{q} = (2{,}1) \to 1$
- $\boldsymbol{q} = (0{,}3) \to -3$
- $\boldsymbol{q} = (0{,}0) \to 0$

The value of the point (4,1) is greater than those of the other points therefore the hyperplane with the normal vector (1, -1) separates that desired point from the others.

(c) The lemma guarantees a positive value by finding a big enough $a \in \mathbb{R}$:

$$a = 2 \implies f_{ex}(a^{(1,-1)}) = f_{ex}(2, \frac{1}{2}) = \frac{29}{2} > 0$$

The intermediate value theorem states that there must exist a root $r \in \overline{\mathbf{1}\boldsymbol{p}}$, where $\boldsymbol{p} = (2, \frac{1}{2})$, of the continuous polynomial function f_{ex}. Therefore is $f_{ex} = 0$ satisfiable. Remember that the subtropical method is incomplete [FOSV17] [Stu15] and that we cannot conclude unsatisfiability in case the method fails to find a positive point.

When the Subtropical Heuristic Succeeds

The subtropical method is incomplete. There are two necessary conditions where at least one needs to be fulfilled for the success of the subtropical method [FOSV17]. This characterization was one of the research goals by Fontaine and colleagues. Identifying the satisfiability of those conditions efficiently could lead to a performance improvement as we either can guarantee to find a solution or cancel this method early and thus saving computation time.

Let $f \in \mathbb{Q}[x_1,...,x_d]$ be a polynomial and $\Pi(f) = \{\boldsymbol{r} \in (0,\infty)^d : f(\boldsymbol{r}) > 0\}$ the set of its positive values. Let also $\overline{\Pi(f)}$ be the topological closure of $\Pi(f)$ which is $\Pi(f)$ itself including limit points of $\Pi(f)$. The two conditions are:

- $\mathbf{0} \in \overline{\Pi(f)}$.
- $\overline{\Pi(f)}$ is unbounded.

These conditions can be inferred from Lemma 2.1.1. For the former condition: If the normal vector of the separating hyperplane consists of only negative values then $a^{\boldsymbol{n}}$ approaches $\mathbf{0}$ for increasing a. Now, because the sign of the polynomial function at $a^{\boldsymbol{n}}$ does not change for all $a \geq a_0$ for some $a_0 \in \mathbb{R}^+ \implies \mathbf{0}$ must belong to $\overline{\Pi(f)}$.
For the other condition: If the normal vector contains at least one positive entry at the index $1 \leq i \leq d$ then the entry at i of $a^{\boldsymbol{n}}$ diverges for increasing a and therefore $\overline{\Pi(f)}$ must be unbounded.

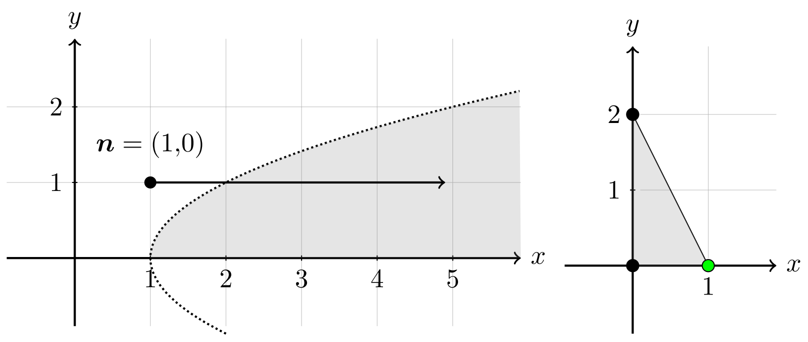

Figure 2.3: $\overline{\Pi(f)}$ and Newton polytope of $f(x,y) = x - y^2 - 1$

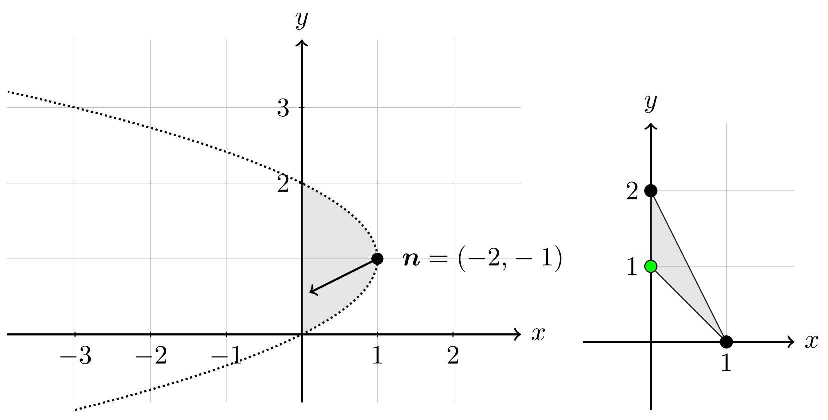

Figure 2.4: $\overline{\Pi(f)}$ and Newton polytope of $f(x,y) = -x - y^2 + 2y$

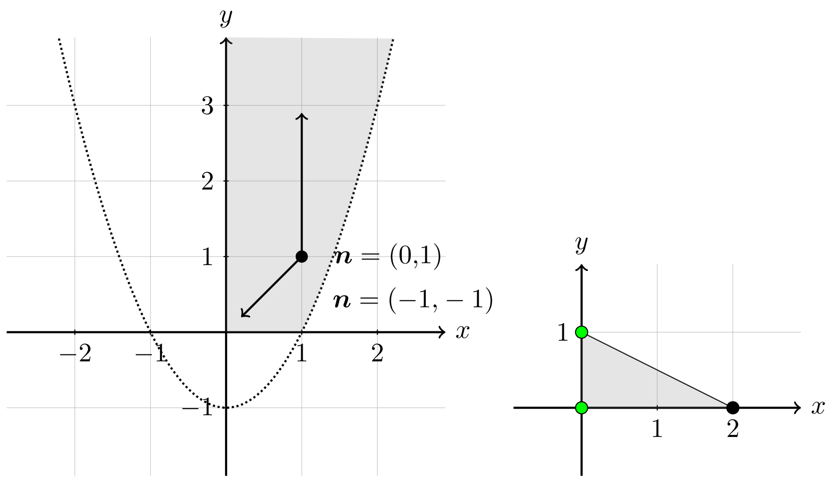

Figure 2.5: $\overline{\Pi(f)}$ and Newton polytope of $f(x,y) = -x^2 + y + 1$

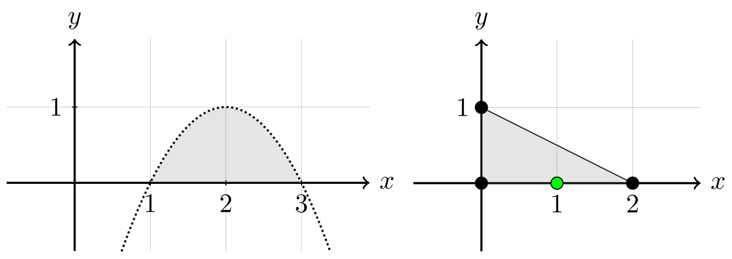

Figure 2.6: $\overline{\Pi(f)}$ and Newton polytope of $f(x,y) = -x^2 + 4x - y - 3$

2.2 Cylindrical Algrebraic Decomposition

In this section we want to introduce cylindrical algebraic decomposition (CAD). The notion and the algorithm for the computation of a CAD was first introduced by George E. Collins in 1975 [Col75]. The idea is that for a given set F of polynomials with d variables we can *decompose* $\mathbb{R}^d$ into regions where each polynomial $f \in F$ is *sign-invariant* for all inputs from that region. Given the combinations of signs each polynomial has in a specific region we can easily determine satisfiability of any polynomial constraint systems that only includes polynomials $f \in F$. This thesis does not cover the complete topic. We present however a broad example to grasp an intuition, give necessary definitions, and show how to compute algebraic solutions for univariate polynomials. For the interested reader: A more detailed introduction can be found in e.g. "Cylindrical algebraic decomposition - an introduction" written by M. Jirstrand, in 1995 [Jir95].

2.2.1 The Intuition for a CAD

Consider the following polynomial constraint system:

$$x^3 - y \sim_1 0$$
$$x^2 + y^2 - 1 \sim_2 0$$

where $\sim_1, \sim_2 \in \{=, <, \leq, >, \geq, \neq\}$. Let $f_1 = x^3 - y$ and $f_2 = x^2 + y^2 - 1$. Figure 2.7 illustrates an input space and shows the roots of those polynomials. Equivalently, if we plot those polynomials as functions in a three dimensional (x,y,z)-coordinate system where z represents the function value then Figure 2.7 would show the segment of the plane at $z = 0$.
Now, since polynomial functions are continuous, the graphs that traces the roots of f_1 and f_2 separate the input space into areas in which the respective polynomial is either only positive or negative.

The projections (shown as green dots in Figure 2.7) we are looking for in this example are indicated by the intersections of the roots of both polynomials or if a vertical tangent encounters a sign change along its line. To elaborate further on the latter:
For $\xi \in \mathbb{R}$ let $f[x/\xi]$ denote the polynomial f when substituting all occurrences of the variable x with the real value ξ. The projection needs to decompose/split the x-axis into regions such that for all regions $R \subseteq \mathbb{R}$ and for each polynomial $f \in \{f_1, f_2\}$ the following holds:

$$\forall \xi, \xi' \in R : |\{y \in \mathbb{R} : (f[x/\xi])(y) = 0\}| = |\{y \in \mathbb{R} : (f[x/\xi'])(y) = 0\}|$$

In other words, the polynomial that we receive after substituting x for any $\xi \in R$ has always the same number of roots. This is among other properties something that characterizes those projections implicitly.
Another property explains the intersection case and characterizes those projections implicitly by having a *constant number of* ***common*** *roots.* Those and another property (which characterizes implicitly projections by having a *constant number of* ***distinct*** *roots*) of a decomposition for a set of polynomials are formalized by the concept of *delineability* which is introduced by M. Jirstrand in [Jir95] chapter 4. The necessity of delineability can be seen when looking at the final results of this example.

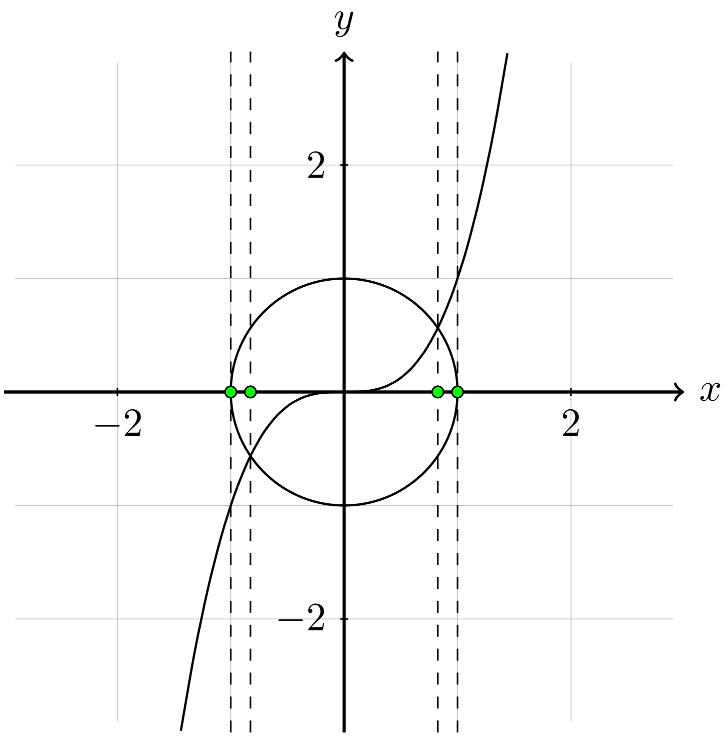

Figure 2.7: Roots of $x^2 + y^2 - 1$ (unit circle) and $x^3 - y$ (cubic function) and projections onto the x-axis.

The projection yields the following values:

$$-1,\ -\alpha,\ \alpha,\ 1 \qquad \text{where } \alpha \approx 0.826$$

With these values we can create a decomposition of the x-axis with 9 regions R_1,...,R_9:

$$\{\underbrace{(-\infty,-1)}_{R_1},\underbrace{\{-1\}}_{R_2},\underbrace{(-1,-\alpha)}_{R_3},\underbrace{\{-\alpha\}}_{R_4},\underbrace{(-\alpha,\alpha)}_{R_5},\underbrace{\{\alpha\}}_{R_6},\underbrace{(\alpha,1)}_{R_7},\underbrace{\{1\}}_{R_8},\underbrace{(1,\infty)}_{R_9}\}$$

The next steps can be broken down to:

- Take one sample point ξ from each region R_1,...,R_9.
- Substitute x in each f_1, f_2 for each sample point ξ to gather in total $2 \cdot 9 = 18$ polynomials.
- Evaluate the sign changes $\begin{pmatrix}(f_1[x/\xi])(y)\\(f_2[x/\xi])(y)\end{pmatrix}$ for $y : -\infty \to \infty$

Table 2.1 concludes these steps:

Region	ξ	$f_1[x/\xi]$	$f_2[x/\xi]$	$\begin{bmatrix} sign((f_1[x/\xi])(y)) \\ sign((f_2[x/\xi])(y)) \end{bmatrix}, \quad y : -\infty \to \infty$
$(-\infty, -1)$	-2	$-y - 8$	$y^2 + 3$	$\begin{bmatrix} + & 0 & - \\ + & + & + \end{bmatrix}$
$\{-1\}$	-1	$-y - 1$	y^2	$\begin{bmatrix} + & 0 & - & - & - \\ + & + & + & 0 & + \end{bmatrix}$
$(-1, -\alpha)$	-0.9	$-y - 0.729$	$y^2 - 0.19$	$\begin{bmatrix} + & 0 & - & - & - & - & - \\ + & + & + & 0 & - & 0 & + \end{bmatrix}$
$\{-\alpha\}$	$-\alpha$	$-y - \alpha^3$	$y^2 + \alpha^2 - 1$	$\begin{bmatrix} + & 0 & - & - & - \\ + & 0 & - & 0 & + \end{bmatrix}$
$(-\alpha, \alpha)$	0.5	$-y + 0.125$	$y^2 - 0.75$	$\begin{bmatrix} + & + & + & 0 & - & - & - \\ + & 0 & - & - & - & 0 & + \end{bmatrix}$
$\{\alpha\}$	α	$-y + \alpha^3$	$y^2 + \alpha^2 - 1$	$\begin{bmatrix} + & + & + & 0 & - \\ + & 0 & - & 0 & + \end{bmatrix}$
$(\alpha,1)$	0.9	$-y + 0.729$	$y^2 - 0.19$	$\begin{bmatrix} + & + & + & + & + & 0 & - \\ + & 0 & - & 0 & + & + & + \end{bmatrix}$
$\{1\}$	1	$-y + 1$	y^2	$\begin{bmatrix} + & + & + & 0 & - \\ + & 0 & + & + & + \end{bmatrix}$
$(1, \infty)$	2	$-y + 8$	$y^2 + 3$	$\begin{bmatrix} + & 0 & - \\ + & + & + \end{bmatrix}$

Table 2.1: Substitutions for each sample point and sign changes for each substitution pair.

The sign pairs in the table can easily identify satisfiability for any given combination $\sim_1, \sim_2 \in \{=, <, \leq, >, \geq, \neq\}$. Consider these examples:

- Suppose we want to solve the polynomial constraint system for $(\sim_1, \sim_2) = (<, <)$. We can see that those constraints are satisfiable using the table if and only if $x \in R_3 \cup R_4 \cup R_5$. We can choose a sample point freely from those regions without changing the sign behavior (this is ensured by delineability). Let $x = 0 \in R_5$ then the system simplifies to:

$$\left.\begin{array}{ll} -y & < 0 \\ y^2 - 1 & < 0 \end{array}\right\} \implies 0 < y < 1$$

- If $(\sim_1, \sim_2) = (>, =)$ then x must be in either of the regions R_5, R_6, R_7 or R_8. Let $x \in R_8 = \{1\}$ then the system simplifies to:

$$\left.\begin{aligned} 1 - y &> 0 \\ y^2 &= 0 \end{aligned}\right\} \implies y = 0$$

The importance of delineability now becomes clear as this concept ensures that for any sample point taken from a specific region the same sign behavior is gathered. We also do not miss any potential solutions.

The decomposition of course is not a decomposition of $\mathbb{R}^2$. In order to achieve this we need to construct the CAD using the information gained from the projections. Explaining this in detail for the general case would exceed the scope of this thesis. Nevertheless, we give a simplistic explanation for our example:
For each region R_i where $1 \leq i \leq 9$ consider the set $R_i' = \{(x,y) \in \mathbb{R}^2 : x \in R_i\}$. Then partition R_i' into non-empty connected sets where each polynomial f_1 and f_2 has a constant sign. Figure 2.8 illustrates those new regions (we omit the axes for better visual perception).

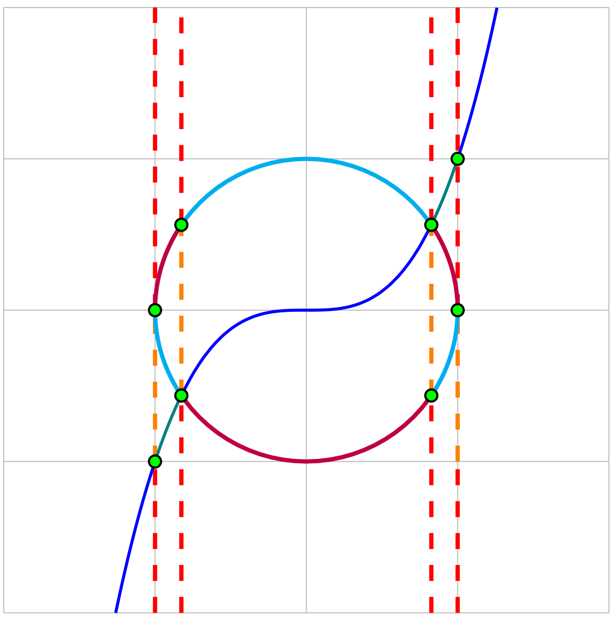

Figure 2.8: CAD of $\{f_1, f_2\}$ with 47 cells (12 line-/6 arc-/5 curve segments, 8 points, and 16 areas).

2.2.2 Terminology

Definition 2.2.1. Algebraic Number

A real (or complex) number α is an *algebraic number* if there exists a univariate polynomial $f \in \mathbb{Q}[x]$ such that:

$$f \neq 0 \wedge f(\alpha) = 0$$

This property with the addition of intervals can be used to represent algebraic numbers. This representation must specify exactly one algebraic number. Therefore the interval must be restricted to contain only one algebraic number for the given polynomial. For example, let $f = x^2 - 2$. The representation of the algebraic numbers $\sqrt{2}$ and $-\sqrt{2}$ are:

$$\sqrt{2} : (f, [1, 1.5]) \quad \text{and} \; -\sqrt{2} : (f, [-1.5, -1])$$

The advantage of using this representation is that we can precisely express numbers which cannot be represented as radicals.
For example, the algebraic number $(x^5 - x - 1, [1, 1.2])$. The existence of such numbers has been proven by Paolo Ruffini and Niels Henrik Abel and is referred to the Abel-Ruffini theorem [Ayo80] [Ruf13] [Abe24] [Abe26].

Definition 2.2.2. Region

A *region* R is a non-empty connected subset of $\mathbb{R}^d$.

Definition 2.2.3. Decomposition

A *decomposition* of $\mathbb{R}^d$ is a finite set $\mathfrak{C} \subseteq 2^{\mathbb{R}^d}$. Its elements are pairwise disjoint regions $C \subseteq \mathbb{R}^d$ when united results in $\mathbb{R}^d$.

Definition 2.2.4. Semi-algebraic

A set is *semi-algebraic* if it is the result of finitely many unions, intersections, and/or complementation on sets of the form:

$$\{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \sim 0\} \quad \text{where } \sim \in \{=, >\} \text{ and } f \in \mathbb{Q}[x_1,...,x_d]$$

A set is *algebraic* if it can be constructed the same way but:

- with the restriction that $\sim \in \{=\}$, i.e. only polynomial equational constraints may be used.
- complex numbers may also be considered.

Definition 2.2.5. Cylindrical

Let $\mathfrak{C}$ of $\mathbb{R}^d$ be a decomposition. If $d = 1$ then $\mathfrak{C}$ is *cylindrical.* If $d > 1$ then let $\mathfrak{C}'$ be the set that results from projecting the regions of $\mathfrak{C}$ to the first $d - 1$ entries. If $\mathfrak{C}'$ is a cylindrical decomposition then $\mathfrak{C}$ is cylindrical.

Definition 2.2.6. Sign-invariance

Let $f \in \mathbb{Q}[x_1,...,x_d]$ and $S \subseteq \mathbb{R}^d$. S is *f-sign-invariant* if:

$$\exists \sim \in \{=, >, <\} \forall \boldsymbol{x} \in S : f(\boldsymbol{x}) \sim 0$$

Analogously, we can define this property for a set of polynomials $F = \{f_1, ..., f_m\} \subset \mathbb{Q}[x_1,...,x_d]$. S is *F-sign-invariant* if S is f-sign-invariant for each $f \in F$.

Definition 2.2.7. Cylindrical Algebraic Decomposition

A *cylindrical algebraic decomposition* of $\mathbb{R}^d$ is a cylindrical and semi-algebraic decomposition of $\mathbb{R}^d$. Its elements are also referred to as *cells.* A CAD of $F \subseteq \mathbb{Q}[x_1,...,x_d]$ is a CAD in that each cell is F-sign-invariant.

2.2.3 Computing Algebraic Real Root Solutions for Univariate Polynomials

In this section we explain how to construct a CAD using *Cauchy bounds* and *Sturm sequences* for a univariate polynomial set.

Cauchy Bound

The *Cauchy bound*, named after Augustin-Louis Cauchy, specifies an upper and lower bound for all roots of a given univariate polynomial $f = \sum_{i=0}^{n} c_i x^i \in \mathbb{Q}[x]$. Let $r \in \mathbb{R}$ be a root of f then [RAUb]:

$$|r| \leq 1 + max\{|\frac{c_0}{c_n}|,|\frac{c_1}{c_n}|, ..., |\frac{c_{n-1}}{c_n}|\}$$

Proof. If $|r| \leq 1$ then the statement is true. Let $|r| > 1$ and $c_{max} = max\{|c_0|,...,|c_{n-1}|\}$ then:

$$f(r) = 0 \iff |c_n \cdot r^n| = |\sum_{i=0}^{n-1} c_i \cdot r^i| \leq \sum_{i=0}^{n-1} |c_i \cdot r^i| \leq \sum_{i=0}^{n-1} |c_{max} \cdot r^n|$$

Factoring c_{max} out and using the formula for a geometric series results in:

$$c_{max} \cdot \sum_{i=0}^{n-1} |r|^n = c_{max} \cdot \frac{|r|^n - 1}{|r| - 1} \leq c_{max} \cdot \frac{|r|^n}{|r| - 1}$$

Finally:

$$\begin{aligned}
|c_n \cdot r^n| &\leq c_{max} \cdot \frac{|r|^n}{|r|-1} && | : |c_n| \\
|r^n| &\leq |\frac{c_{max}}{c_n} \cdot \frac{|r|^n}{|r|-1}| && | : |r^n| \\
1 &\leq |\frac{c_{max}}{c_n} \cdot \frac{1}{|r|-1}| && | \cdot (|r|-1) \\
|r| - 1 &\leq |\frac{c_{max}}{c_n}| && | + 1 \\
|r| &\leq 1 + |\frac{c_{max}}{c_n}|
\end{aligned}$$

□

Sturm sequence

Let $f \in \mathbb{Q}[x]$ be a square-free polynomial (i.e. the number of roots f has is equal to the number of distinct roots f has). The *Sturm sequence*, named after Jacques Charles François Sturm, is a sequence of univariate polynomials $f_0,f_1,...,f_n \in (\mathbb{Q}[x])^{n+1}$ that allows us to count the number of roots f in a real interval $(a,b]$ has [RAUb] [Rag] [Stu09]. It is inductively defined using a derivative and a variant of the Euclidean algorithm:

1. $f_0 := f, \quad f_1 := f' = \frac{d}{dx} f.$
2. For $i \geq 2$ is $f_i := -\text{rem}(f_{i-2}, f_{i-1})$ where $\text{rem}(f_{i-1}, f_{i-2})$ is the remainder of f_{i-1} divided by f_{i-2}. The sequence stops as soon as $-\text{rem}(f_{i-2}, f_{i-1}) = 0$ then $f_n := f_{i-1}$.

The construction of these polynomials fulfill certain properties which ensures the correctness of the following theorem [Rag].

Theorem 2.2.1. Sturm's Theorem

For a given Sturm sequence $f_0, ..., f_n \in (\mathbb{Q}[x])^{n+1}$ of a square-free polynomial f let $\sigma(\xi)$ count the number of *sign changes* in

$$(f_0(\xi), ..., f_n(\xi))$$

ignoring zeros. Then the number of distinct real roots f in a real interval $(a,b]$ has is equal to [RAUb] [Rag] [Stu09]:

$$\sigma(a) - \sigma(b)$$

We can extend this theorem for unbounded intervals. Let $LC_0, ..., LC_n$ denote the leading coefficients and $LE_0, ..., LE_n$ the exponents of the leading terms of each polynomial $f_0, ..., f_n$ respectively. Then define

- $\sigma(\infty)$ to count the number of sign changes in $(\text{sign}(LC_0), ..., \text{sign}(LC_n))$ and
- $\sigma(-\infty)$ to count the number of sign changes in $(\text{sign}(LC_0(-1)^{LE_0}), ..., \text{sign}(LC_n(-1)^{LE_n}))$.

Then the number of roots f has is equal to $\sigma(-\infty) - \sigma(\infty)$.

The following proof uses and builds on the proof given in [Rag]. Consider these statements about the Sturm sequence:

1. No consecutive polynomials have a common root:
$$\forall 0 \leq i \leq n : f_i(r) = 0 \implies f_{i+1}(r) \neq 0$$
2. If a polynomial in the sequence other than f_0 or f_n has a root r then its successor and predecessor have opposite signs at r:
$$\forall 0 < i < n : f_i(r) = 0 \implies (\text{sign}(f_{i-1}(r)), \text{sign}(f_{i+1}(r))) \in \{(+,-), (-,+)\}$$
3. $f_n \neq 0$ has a constant sign along its domain:
$$\forall x \in \mathbb{R} : f_n(x) > 0 \lor f_n(x) < 0$$

We first prove them:
Statement 1):

$i = 0$ f is square-free and has therefore no common roots with f' [sul15].

$i > 0$ Suppose f_i and f_{i+1} have a common root r and let g_1, g_2 be two polynomials such that $f_i = g_1(x-r)$ and $-f_{i+1} = g_2(x-r)$. However:

$$f_{i-1} = g_1(x-r) + g_2(x-r) = (g_1 + g_2)(x-r)$$

This implies that r is also a common root of f_{i-1}. This is however a contradiction if $i = 1$. Therefore f_1 and f_2 cannot share common roots and by induction the same contradiction occurs for all $i > 0$.

Statement 2):
Let r be the root of f_i for $0 < i < n$ then we have:

$$f_{i-1}(r) = g(r) \cdot \underbrace{f_i(r)}_{=0} + (-f_{i+1}(r)) = -f_{i+1}(r)$$

because no consecutive polynomials in the sequence share a root f_{i-1} and f_{i+1} are not 0 at r and therefore have opposite signs at r.

Statement 3):
Suppose $f_n \neq 0$ has not the same sign over its domain then it must have at least one root r and we can write f_n as $g(x-r)$ where $g \in \mathbb{Q}[x]$. Since f_n is a greatest common divisor of f and f' we can write them as:

$$f = g_1(x-r), \quad f' = g_2(x-r)$$

which is a contradiction to 1). Therefore f_n must have a constant sign over its domain.

We claim an equivalent statement to the theorem: "*For a given Sturm sequence of a univariate polynomial f, $\sigma(x)$ decreases if and only if $f(x) = 0$*". The number of sign changes at $\sigma(x)$ may only change if an f_i in the sequence changes its sign near the neighborhood of its roots. Consider the following cases:

- r is a root of $f_0 = f$ then there exists a small enough $0 < \epsilon < 1$ such that $f(r-\epsilon)$ and $f(r+\epsilon)$ have opposite signs since f is square-free. Assume $f(r-\epsilon) > 0$ then $f'(r-\epsilon)$ must be negative as f approaches 0 "from above the x-axis". Therefore f is decreasing within the interval $[r-\epsilon, r]$. Now because $f_1 = f'$, observing the course of the signs of the Sturm sequence from $r-\epsilon$ to $r+\epsilon$ yields:

$$\underbrace{(+, -, ...)}_{r-\epsilon} \rightarrow \underbrace{(0, -, ...)}_{r} \rightarrow \underbrace{(-, -, ...)}_{r+\epsilon}$$

  $\sigma(x)$ decreases exactly by one at each root r of f. The observation for $f(r-\epsilon) < 0$ is analogous.

- r is a root of f_i for an $0 < i < n$. Similarly, there exists an $0 < \epsilon < 1$ such that $f_i(r-\epsilon)$ and $f_i(r+\epsilon)$ have opposite signs. Statement 2) implies that $f_{i-1}(r) \cdot f_{i+1}(r) < 0$, i.e. those polynomials do not have the same sign at r. Observing the course of the signs of the Sturm sequence from $r-\epsilon$ to $r+\epsilon$ yields multiple cases (we show only the signs of f_{i-1}, f_i, and f_{i+1} in the sequence):

  - $f_{i-1}(r) < 0$ and $f_i(r-\epsilon) < 0$:

$$\underbrace{(.., -, -, + ..)}_{r-\epsilon} \rightarrow \underbrace{(.., -, 0, + ..)}_{r} \rightarrow \underbrace{(.., -, +, + ..)}_{r+\epsilon}$$

- $f_{i-1}(r) < 0$ and $f_i(r-\epsilon) > 0$:

$$\underbrace{(..,-\,,+\,,+\,..)}_{r-\epsilon} \rightarrow \underbrace{(..,-\,,0,+\,..)}_{r} \rightarrow \underbrace{(..,-\,,-\,,+\,..)}_{r+\epsilon}$$

- $f_{i-1}(r) > 0$ and $f_i(r-\epsilon) < 0$:

$$\underbrace{(..,+\,,-\,,-\,..)}_{r-\epsilon} \rightarrow \underbrace{(..,+\,,0,-\,..)}_{r} \rightarrow \underbrace{(..,+\,,+\,,-\,..)}_{r+\epsilon}$$

- $f_{i-1}(r) > 0$ and $f_i(r-\epsilon) > 0$:

$$\underbrace{(..,+\,,+\,,-\,..)}_{r-\epsilon} \rightarrow \underbrace{(..,+\,,0,-\,..)}_{r} \rightarrow \underbrace{(..,+\,,-\,,-\,..)}_{r+\epsilon}$$

We observe that for each case that $\sigma(x)$ does not decrease nor increase.

- Statement 3) $\implies$ f_n has no real roots and has no influence on σ.

Statement 1) ensures that no consecutive polynomials in the sequence can simultaneously change their signs so those cases can be neglected. □

Example 2.2.1. Creating a CAD for a simple univariate polynomial constraint set.

Let $f := x^2 - x - 6$ and $g := 2x - 2$. Their respective Cauchy bounds are 7 and 2. Their Sturm sequences are $(x^2 - x - 6, 2x - 1, \frac{25}{4})$ and $(2x - 2, 2)$ respectively. The following table shows the sign sequences along some given points:

x	Signs of the Sturm sequence of f	$\sigma(x)$	Signs of the Sturm sequence of g	$\sigma(x)$
-7	$(+,-,+)$	2	$(-,+)$	1
-2	$(0,-,+)$	1	$(-,+)$	1
0	$(-,-,+)$	1	$(-,+)$	1
2	$(-,-,+)$	1	$(+,+)$	0
7	$(+,+,+)$	0	$(+,+)$	0

Table 2.2: Signs of the Sturm sequence of f and g at given points and their number of sign changes

f has in total $\sigma(-7) - \sigma(7) = 2$ roots and g has $\sigma(-2) - \sigma(2) = 1$ root. The roots of f lie in the intervals $[-7{,}0]$ and $[0{,}7]$ and the root of g in $[-2{,}2]$. We can depict these roots using the representation for algebraic numbers:

$$\alpha := (f, [-7{,}0]), \quad \beta := (g, [-2{,}2]), \quad \gamma := (f, [0{,}7])$$

We can see from the σ-entries in table 2.2 that $\alpha < \beta < \gamma$. The CAD for a single univariate polynomial can be easily constructed as they are exactly the sign-invariant intervals or singletons for that polynomial:

- The CAD for $\{f\}$ is $\text{CAD}_f = \{(-\infty,\alpha), \{\alpha\}, (\alpha,\gamma), \{\gamma\}, (\gamma,\infty)\}$.

- The CAD for $\{g\}$ is $\mathrm{CAD}_g = \{(-\infty, \beta), \{\beta\}, (\beta, \infty)\}$.

The CAD for $\{f,g\}$ is the set that contains the *non-empty intersections* of each element in CAD_f with each element in CAD_g or formally expressed:

$$\mathrm{CAD}_{f,g} = \{(I_1 \cap I_2) \neq \emptyset : I_1 \in \mathrm{CAD}_f \wedge I_2 \in \mathrm{CAD}_g\}$$

The enumeration of $\mathrm{CAD}_{f,g}$ is:

$$\{(-\infty,\alpha), \{\alpha\}, (\alpha, \beta), \{\beta\}, (\beta, \gamma),\{\gamma\}, (\gamma, \infty)\}$$

Chapter 3

The Transformation and Equisatisfiability

The main focus of this research is to analyze and implement a transformation, suggested by Dr. Hamid Rahkooy, that should map polynomial constraints, their negation, their disjunction, and their conjunction to equisatisfiable equations. The ultimate goal is to achieve a single (quantifier-free) polynomial equational constraint whose satisfiability could be determined using the subtropical real root finding method.

In this chapter we introduce that transformation and give some elaboration. We also give a short overview of some necessary terminology. Let φ, ψ be formulas:

- φ, ψ are *logically equivalent* if every model that satisfies φ also satisfies ψ and vice versa. We denote this relation by $\varphi \equiv \psi$.
- ψ is a *logical consequence* of φ if every model that satisfies φ also satisfies ψ. We denote this relation by $\varphi \vDash \psi$.
- φ, ψ are *equisatisfiable* when φ is satisfiable if and only if ψ is satisfiable. We denote this relation by $\varphi \equiv_{sat} \psi$.

Let $f, g \in \mathbb{Q}[x_1,...,x_d]$ be polynomials. The transformation t is defined as follows:

1. $t(\neg(f \sim 0)) := t(f \not\sim 0)$ where $\sim \in \{=, <, \leq, >, \geq, \neq\}$ and:
   - If $\sim \in \{=\}$ then $\not\sim \in \{\neq\}$
   - If $\sim \in \{<\}$ then $\not\sim \in \{\geq\}$
   - If $\sim \in \{\leq\}$ then $\not\sim \in \{>\}$
   - If $\sim \in \{>\}$ then $\not\sim \in \{\leq\}$
   - If $\sim \in \{\geq\}$ then $\not\sim \in \{<\}$
   - If $\sim \in \{\neq\}$ then $\not\sim \in \{=\}$
2. $t(f = 0) := (f = 0)$
3. $t(f < 0) := (y^2 f + 1 = 0)$ where y is a fresh new variable, i.e. does not occur as variable in f. The same holds for the transformation rules 4 to 7.

4. $t(f > 0) := (y^2 f - 1 = 0)$

5. $t(f \leq 0) := (f + y^2 = 0)$

6. $t(f \geq 0) := (f - y^2 = 0)$

7. $t(f \neq 0) := (yf + 1 = 0)$

8. $t(f = 0 \vee g = 0) := (fg = 0)$

9. $t(f = 0 \wedge g = 0) := t(\neg(t(t(f \neq 0) \vee t(g \neq 0))))$

We can generalize rule 8 and 9 for $m > 1$:

- Disjunction: $t(\bigvee_{i=1}^{m} f_i = 0) := (\prod_{i=1}^{m} f_i = 0)$

- Conjunction: $t(\bigwedge_{i=1}^{m} f_i = 0) := t(\neg(t(\bigvee_{i=1}^{m} t(\neg f_i = 0)))$

The transformation rule 9 for two constraints fully evaluated results in:

$$\begin{aligned} t(f = 0 \wedge g = 0) &= t(\neg(t(t(f \neq 0) \vee t(g \neq 0)))) && \text{Rule 9} \\ &= t(\neg(t(y_1 f + 1 = 0 \vee y_2 g + 1 = 0))) && \text{Rule 7} \\ &= t(\neg(y_1 f + 1)(y_2 g + 1) = 0)) && \text{Rule 8} \\ &= t((y_1 f + 1)(y_2 g + 1) \neq 0)) && \text{Rule 1} \\ &= y_3 (y_1 f + 1)(y_2 g + 1) + 1 = 0 && \text{Rule 7} \end{aligned}$$

Similarly, the transformation of $m > 1$ constraints yields:

$$t(\bigwedge_{i=1}^{m} f_i = 0) = y_{m+1}(\prod_{i=1}^{m}(y_i f_i + 1)) + 1 = 0$$

3.1 Transformation Properties

3.1.1 Termination

An algorithm that uses the proposed transformation halts on every correct input. The following recursive pseudo-algorithm exemplifies this.

Algorithm 1 Pseudo code of the transformation

Input QF NRA formula φ

Output Single polynomial equational constraint

```
1: if $\varphi$ is not in CNF then
2:     $\varphi' \leftarrow \varphi$ in CNF.
3:     return call transformation on $\varphi'$
4: end if
5: if $\varphi$ is a literal then
6:     if $\varphi$ has negation then
7:         $\varphi' \leftarrow$ move negation into relation of $\varphi$
8:         return call transformation on $\varphi'$
9:     else
10:        $f \leftarrow$ polynomial in $\varphi$
11:        $y \leftarrow$ fresh new variable
12:        switch Relation of $\varphi$ do
13:            case EQUAL
14:                return $\varphi$
15:            case LESS
16:                return $y^2 \cdot f + 1 = 0$
17:            case LESS_OR_EQUAL
18:                return $f + y^2 = 0$
19:            case GREATER
20:                return $y^2 \cdot f - 1 = 0$
21:            case GREATER_OR_EQUAL
22:                return $f - y^2 = 0$
23:            case NOT_EQUAL
24:                return $y \cdot f + 1 = 0$
25:    end if
26: end if
27: if $\varphi$ is a disjunction then
28:    $L \leftarrow$ literals in $\varphi$
29:    $f \leftarrow 1$
30:    while $L \neq \emptyset$ do
31:        $\psi \leftarrow$ Element of $L$
32:        $L \leftarrow L \setminus \{\psi\}$
33:        $\psi' \leftarrow$ transformation on $\psi$
34:        $f \leftarrow f \cdot$ ( polynomial in $\psi'$)
35:    end while
36:    return $f = 0$
37: end if
38: if $\varphi$ is a conjunction then
39:    $D \leftarrow$ disjunction in $\varphi$
40:    $f \leftarrow 1$
41:    while $D \neq \emptyset$ do
42:        $\psi \leftarrow$ Element of $D$
43:        $\psi' \leftarrow$ transformation on $\psi$
44:        $g \leftarrow$ polynomial in $\psi'$
45:        $y' \leftarrow$ fresh new variable
46:        $f \leftarrow f \cdot (y' \cdot g + 1)$
47:        $D \leftarrow D \setminus \{\psi\}$
48:    end while
49:    $y \leftarrow$ fresh new variable
50:    return $y \cdot f + 1 = 0$
51: end if
```

Proof of termination. Let φ be a quantifier-free non-linear real arithmetic formula. Distinguish the following cases:

- φ is not in CNF. Now, since φ does not contain any quantifier it is similarly structured like a propositional formula. Therefore exists a formula ψ such that:

  $$\psi \text{ is in CNF and } \psi \equiv \varphi$$

  The next call of the function is guaranteed to fall under one of the following cases.

- φ is a literal. If φ is a negation of an atomic formula then the same algorithm is called on the adjusted formula which is an atomic formula with the same left- and right hand side and its relation changed accordingly.
  Otherwise, φ is an atomic formula and the function returns a single polynomial constraint. The algorithm terminates on literals.

- φ is a disjunction of literals. The loop is bounded by the constant amount of literals φ has. The argument in the function call within the loop is a literal and terminates therefore as well. Therefore the algorithm terminates on disjunctions of literals.

- φ is a conjunction of disjunctions of literals. The loop is bounded by the constant amount of disjunctions φ has. Analogously, the function call within the loop has a disjunction of literals as argument and terminates therefore as well. Finally, the algorithm terminates for every QF NRA formula.

3.1.2 Equisatisfaction

The transformation needs to be equisatisfiable to its input in order to determine satisfiability. The given table below is a quick overview for the its relation to the original formula (we assume for rule 9 that the transformation is fully evaluated).

φ	$t(\varphi) \equiv_{sat} \varphi$	$t(\varphi) \vDash \varphi$	$\varphi \vDash t(\varphi)$	$\varphi \equiv t(\varphi)$
$f = 0$	Yes	Yes	Yes	Yes
$f < 0$	Yes	Yes	No	No
$f > 0$	Yes	Yes	No	No
$f \leq 0$	Yes	Yes	No	No
$f \geq 0$	Yes	Yes	No	No
$f \neq 0$	Yes	Yes	No	No
$f = 0 \vee g = 0$	Yes	Yes	Yes	Yes
$f = 0 \wedge g = 0$	No	No	No	No

Table 3.1: Relations to the transformation

The first transformation rule is omitted because the act of moving the negation inside the relation yields a logically equivalent formula and therefore we need to consider the other rules only.

The reason why rule 2 fulfills all relations is trivial since that transformation rule is the identity.
The transformations of rule 3 to rule 7 are not logically equivalent as they introduce

new variables. A model that satisfy the original formula does not necessarily satisfy the transformation as we can choose the freshly introduced variable freely such that the transformation can be falsified. However, they are equisatisfiable which is the necessary trait for determining satisfiability.
Transformation rule 8 on the other hand is logically equivalent because the reals with its addition and multiplication is a field. This means that it's also an integral domain and therefore fulfills the following condition:

$$\forall a,b \in \text{Domain} : (a \neq 0 \wedge b \neq 0 \iff a \cdot b \neq 0)$$

which is equivalent to:

$$\forall a,b \in \text{Domain} : (a = 0 \vee b = 0 \iff a \cdot b = 0)$$

The transformation rule 9 is the only one that does not return an equisatisfiable formula (this means that the constraint returned by the pseudo-algorithm is in general not equisatisfiable). Consider the following counterexample.

Let $m > 1$ and $f_1,...,f_m \in \mathbb{Q}[x_1,...,x_d]$ be polynomials such that $\bigwedge_{i=1}^m f_i = 0$ is unsatisfiable. Then

$$t(\bigwedge_{i=1}^{m} f_i = 0) = y_{m+1}(\prod_{i=1}^{m}(y_i f_i + 1)) + 1 = 0, \quad y_1,...,y_{m+1} \text{ are fresh variables}$$

can be satisfied by $y_1 = ... = y_m = 0$ and $y_{m+1} = -1$. The other variables can be chosen at random as they do not matter. In conclusion, even if $\bigwedge_{i=1}^m f_i = 0$ is unsatisfiable the transformation is not.

Notice that we cannot add more restrictions (i.e. constraints) to the fresh real variables. Otherwise, we would not receive a single constraint. This is because they would need to be considered as well in the transformation. This would lead back to the original problem of having multiple constraints.

3.2 Further Analysis

The transformation rule 9 seems at the first glance correct. It utilizes a variation of De Morgan's law and under the assumption that the other transformations are correct, it should return an equisatisfiable formula as well. This fallacy occurs since the following was overseen:

$$\neg\forall\varphi,\psi \in \text{FO}(\{0,1,+,\cdot,<\}) : \varphi \equiv_{sat} \psi \iff \neg\varphi \equiv_{sat} \neg\psi$$

In other words, if two formulas are equisatisfiable then their negation does not have to be as well. This equivalence only holds true if:

- φ and ψ are a tautology.
- φ and ψ are unsatisfiable.
- $\varphi, \psi, \neg\varphi$, and $\neg\psi$ are satisfiable.

Therefore even if $t(\varphi) \equiv_{sat} \varphi$ then $\neg t(\varphi) \equiv_{sat} \neg\varphi$ is not necessarily true. Another example is the constraint $\neg(x^2 + 1 \geq 0)$ which is unsatisfiable. However:

$$\neg(t(x^2 + 1 \geq 0)) = \neg((x^2 + 1) - y^2 = 0) \equiv (x^2 + 1 - y^2 \neq 0)$$

can be satisfied by the interpretation $x = y = 0$.

3.2.1 Alternative Ideas

A correct transformation is by no means impossible since

$$\bigwedge_{i=1}^{m} (f_i = 0) \equiv \sum_{i=1}^{m} f_i^2 = 0$$

however, such a transformation renders the subtropical method less useful because that polynomial has no negative points [1]. Essentially, a complete and correct transformation of a polynomial constraint set Φ must yield a polynomial f that fulfills the following conditions for the application of the subtropical real root finding method:

- $f = 0 \equiv_{sat} \bigwedge \Phi$
- If $\bigwedge \Phi$ is satisfiable then there exists $\boldsymbol{p}$ and $\boldsymbol{q}$ such that $f(\boldsymbol{p}) > 0$ and $f(\boldsymbol{q}) < 0$.

There is also the alternative to allow for quantifiers. The following transformation also yields a logically equivalent (and therefore equisatisfiable) formula:

$$\bigwedge_{i=1}^{m} (f_i = 0) \equiv \forall y : ((\prod_{i=1}^{m} (y \cdot f_i + 1)) - 1 = 0)$$

Proof. Let $\varphi = \bigwedge_{i=1}^{m} (f_i = 0)$ and $\psi = \forall y : ((\prod_{i=1}^{m} (y \cdot f_i + 1)) - 1 = 0)$. Consider the following cases:

- Claim: $\varphi \vDash \psi$
  Let M be a model of φ (or $M \vDash \varphi$ for short). Substituting the free variables $\boldsymbol{x} = (x_1,...,x_n)$ in ψ with the interpretation of M we get:

$$\begin{aligned}
\psi[\boldsymbol{x}/M(\boldsymbol{x})] &= (\forall y : (\prod_{i=1}^{m} (y \cdot f_i + 1)) - 1 = 0)[\boldsymbol{x}/M(\boldsymbol{x})] \\
&\vdash \forall y : (\prod_{i=1}^{m} (y \cdot 0 + 1)) - 1 = 0 \\
&\vdash \forall y : (\prod_{i=1}^{m} (0 + 1)) - 1 = 0 \\
&\vdash \forall y : (\prod_{i=1}^{m} 1) - 1 = 0 \\
&\vdash \forall y : 1 - 1 = 0 \\
&\vdash \forall y : 0 = 0
\end{aligned}$$

  $0 = 0$ is a true sentence and y does not occur in $0 = 0$ therefore $M \vDash \psi$.

[1]but by further transformation due to other constraints this might change

- Claim: $\psi \vDash \varphi$
Let $M \vDash \psi$ and $c_i = f_i[\boldsymbol{x}/M(\boldsymbol{x})]$ for each $1 \leq i \leq m$. Consider the following interpretation:

$$\begin{aligned}\psi[\boldsymbol{x}/M(\boldsymbol{x})] &= (\forall y : (\prod_{i=1}^{m}(y \cdot f_i + 1)) - 1 = 0)[\boldsymbol{x}/M(\boldsymbol{x})] \\ &\vdash \forall y : (\prod_{i=1}^{m}(y \cdot c_i + 1)) - 1 = 0 \\ &\equiv \forall y : ((\prod_{i=1}^{m} c_i) \cdot y^m + ... + (\sum_{i=1}^{m} c_i) \cdot y + 1) - 1 = 0 \\ &\vdash \forall y : (\prod_{i=1}^{m} c_i) \cdot y^m + ... + (\sum_{i=1}^{m} c_i) \cdot y = 0\end{aligned}$$

The polynomial in that formula is equivalent to a univariate polynomial with the indeterminate y with real coefficients. The semantic statement of the formula is "That polynomial is the zero polynomial" which means that all coefficients must be zero. We prove now that $M \vDash \varphi$ by contradiction. Suppose $M \nvDash \varphi$ then there exist non-zero $c_{i_1},...,c_{i_k}$ such that each c_{i_l} belongs to a different polynomial f_{i_l} for $1 \leq l \leq k \leq m$. Now consider the coefficient at y^k:

  - $k = m$
  Then the coefficient $\prod_{i=1}^{m} c_i \neq 0$ (because $\mathbb{R}$ is an integral domain) which is a contradiction to the statement of the formula.
  - $k < m$
  Let I_k denote the set containing all combinations of k indices $(i_1,...,i_k)$ from a set $\{1,...,m\}$ and let $(j_1,...,j_k) \in I_k$ be the indices where $c_{j_1} \neq 0,..., c_{j_k} \neq 0$. Then the coefficient at y^k is:

  $$\sum_{(i_1,...,i_k) \in I_k} (c_{i_1} \cdot ... \cdot c_{i_k})$$

  Each term in that sum that contains at least one c_i with $i \notin \{j_1,...,j_k\}$ vanishes and the coefficient simplifies to:

  $$c_{j_1} \cdot ... \cdot c_{j_k} \neq 0$$

  which is a contradiction to the statement of the formula.

Therefore $M \vDash \varphi$. □

3.3 Model Construction for a Polynomial Equation

Within this section we present a recipe how to construct a model for a polynomial constraint set Φ, assuming that we do gather a single satisfiable quantifier-free polynomial equation φ such that $\varphi \vDash \bigwedge \Phi$.

Let the above mentioned assumptions hold and $\varphi = (f = 0)$. Let also $\boldsymbol{p}$ be a point, which is computed using the subtropical method, such that $f(\boldsymbol{p}) > 0$ and $f(\mathbf{1}) < 0$:

1. Construct a univariate polynomial $f_u(\lambda)$ which is a restriction of f in form of a linear combination between $f(\mathbf{1})$ and $f(\boldsymbol{p})$:

   $$f_u(\lambda) := f(\lambda_1,\ldots,\lambda_n) \quad \lambda_i := (1-\lambda) + \lambda p_i \text{ for } 1 \le i \le m$$

   The domain of f_u is [0,1] and we have $f_u(0) = f(\mathbf{1})$ and $f_u(1) = f(\boldsymbol{p})$.

2. Proceed to solve the univariate polynomial using the CAD method from *Section 2.2.3* to find a real root r which may be represented as an algebraic number $(f_u, [a,b])$ of f_u. Notice that we cannot use the $(f_u, [0,1])$ as a solution because we cannot guarantee that f_u has only one root in that interval.

The model of Φ is then the solution to the simpler polynomial constraint system:

$$\begin{aligned} \lambda - a &\ge 0 \\ \lambda - b &\le 0 \\ f_u &= 0 \\ x_1 - \lambda_1 &= 0 \\ \vdots\ &= 0 \\ x_m - \lambda_m &= 0 \end{aligned}$$

The polynomials in this system have at most 2 variables and the all of them are linear with the only exception being the univariate polynomial which is in general not bounded to be linear. Equivalently, we can also represent the model using expressions whose values are dependent on the substitution used:

$$\forall 1 \le i \le m : x_i = \lambda_i[\lambda/r] \quad \text{where } r = (f_u, [a,b])$$

3.4 Outlook

We reiterate our results in this section and present a couple of outlooks.

3.4.1 Gröbner Basis

It's possible to transform a given polynomial constraint set Φ into an equisatisfiable polynomial equation set Φ_{EQ}. In general, Φ_{EQ} introduces new variables and has a greater degree than within its elements than Φ. Although, Φ_{EQ} in turn can be simplified again using a Gröbner basis [BK10]. The Gröbner basis is a generating set $G = (g_1,\ldots,g_m) \subseteq \mathbb{Q}[x_1,\ldots,x_d]$ of an ideal $I \subseteq \mathbb{Q}[x_1,\ldots,x_d]$ if each $f \in I$ can be represented as a linear combination using G and polynomials $h_1,\ldots,h_m \in \mathbb{Q}[x_1,\ldots,x_d]$:

$$\forall f \in I : f = \sum_{i=1}^{m} h_i \cdot g_i$$

A Gröbner basis can be computed with Buchberger's algorithm [oM] which also can be used to construct a Gröbner basis for Φ_{EQ}. The Gröbner basis $G = (g_1,\ldots,g_m)$ of Φ_{EQ} may induce a simplified polynomial equation set Φ_G with less degree and less amount of variables (not to be confused with less variables). The solution to Φ_G is exactly the solution to Φ_{EQ} because for each polynomial f with $\varphi = (f = 0) \in \Phi_{EQ}$ holds:

$$f = \sum_{i=1}^{m} h_i \cdot \underbrace{g_i}_{=0} = 0$$

Finally, since each formula $\varphi \in \Phi$ is a logical consequence of $\bigwedge \Phi_{EQ}$ the model that fulfills Φ_G and therefore Φ_{EQ} also satisfies Φ.

3.4.2 Gradient Equation Solving

Consider the following transformation for $m > 1$ again:

$$\bigwedge_{i=1}^{m} (f_i = 0) \equiv \sum_{i=1}^{m} f_i^2 = 0$$

As we already mentioned, the subtropical real root finding method is always unsuccessful for such a constraint. Instead, we could attempt to solve and evaluate the gradient of that transformation. The root $\boldsymbol{r}$ of a non-negative polynomial f is a global minimum because it fulfills:

$$\forall \boldsymbol{x} \in \mathrm{dom}(f) : f(\boldsymbol{r}) = 0 \leq f(\boldsymbol{x})$$

A necessary condition for a global minimum $\boldsymbol{m}$ is that the gradient ∇f of a given polynomial f evaluates to the null vector $\mathbf{0}$ at $\boldsymbol{m}$ or simply put:

$$\nabla f(\boldsymbol{m}) = \mathbf{0}$$

We can determine unsatisfiability of a polynomial constraint set Φ if the polynomial equation system induced from ∇f, where $\varphi = (f = 0)$ and $\varphi \vDash \bigwedge \Phi$, has no solutions.

Proof. Let Φ be a polynomial constraint set such that $m = |\Phi| > 1$, i.e. we have at least one conjunction. Let also $\varphi_i = (f_i = 0)$ be the transformed constraints for $1 \leq i \leq m$ and $f \in \mathbb{Q}[x_1,...,x_d]$ the above mentioned transformation of the conjunction of $f_1,...,f_m$:

$$f = \sum_{i=1}^{m} f_i^2$$

If the polynomial equation system gathered from ∇f has no solutions then:

$$\begin{aligned} & \forall \boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \neq 0 \\ \Longrightarrow\ & \exists 1 \leq i \leq m \forall \boldsymbol{x} \in \mathbb{R}^d : f_i^2(\boldsymbol{x}) > 0 \\ \Longrightarrow\ & \exists 1 \leq i \leq m \forall \boldsymbol{x} \in \mathbb{R}^d : f_i(\boldsymbol{x}) > 0 \\ \Longrightarrow\ & \bigwedge_{i=1}^{m} (f_i = 0) \text{ is unsatisfiable} \\ \Longrightarrow\ & \Phi \text{ is unsatisfiable} \end{aligned}$$

□

The converse of this condition is not true. A global minimum does not necessarily have to be a root of f. Take for example $x^2 + 1$, it has no roots but a global minimum at 0. Another example is $g = 2x^4 + 6x^2 + 5 = (x^2 + 1)^2(x^2 + 2)^2$. Its gradient or rather derivative is $g' = 8x^3 + 12x$ which has a root at 0, however, g has no roots.

There is one major downside to this approach. Going from determining satisfiability of a polynomial constraint set to attempting to solve the polynomial equation system induced by the gradient is by no means a reduction to a simpler problem. It rather becomes even more difficult as we introduce more variables which increase the number of entries in the gradient. The following constraint system exemplifies this:

$$\begin{aligned} x_1 + x_2 &> 0 \\ x_2 + 5 &< 0 \end{aligned}$$

The complete transformation with variable ordering $x_1 < x_2 < y_1 < y_2$ to a single polynomial yields:

$$x_1^2y_1^4 + 2x_1x_2y_1^4 - 2x_1y_1^2 + x_2^2y_1^4 + x_2^2y_2^4 - 2x_2y_1^2 + 10x_2y_2^4 + 2x_2y_2^2 + 25y_2^4 + 10y_2^2 + 2$$

The corresponding polynomial equation system induced by the gradient of that transformed polynomial is:

$$\begin{aligned} 2x_1y_1^4 + 2x_2y_1^4 - 2y_1^2 &= 0 \\ 2x_1y_1^4 + 2x_2y_1^4 + 2x_2y_2^4 - 2y_1^2 + 10y_2^4 + 2y_2^2 &= 0 \\ 4x_1^2y_1^3 + 8x_1x_2y_1^3 - 4x_1y_1 + 4x_2^2y_1^3 - 4x_2y_1 &= 0 \\ 4x_2^2y_2^3 + 40x_2y_2^3 + 4x_2y_2 + 100y_2^3 + 20y_2 &= 0 \end{aligned}$$

3.4.3 Quantifier Elimination

There are methods to eliminate quantifiers in non-linear real arithmetic formulas and receive equivalent QF NRA formulas [Neu18]. Regarding the main goal of this thesis, we cannot make use of such quantifier elimination for QF NRA formulas. The reason for this is that this procedure does not guarantee to not introduce any conjunctions, disjunctions, negations, or changes of relations. For example the *parametric parabola problem* [Neu18] is a quantified equational constraint:

$$\exists x(c + bx + ax^2 = 0)$$

The QEPCAD quantifier elimination described in the bachelor thesis [Neu18] by T. Neuhäuser returns an equivalent quantifier-free formula but also discards the "single-equation-constraint" property:

$$(ac - b^2 \leq 0) \wedge (c = 0 \vee a \neq 0 \vee 4ac - b^2 < 0)$$

3.4.4 NRA SMT solvers

Although the subtropical method in its current state cannot handle general NRA formulas that contain quantifiers other SMT solvers that benefit from less formulas and are sturdy against a large number of variables may improve on that.

Chapter 4

Utility for SMT-RAT

SMT-RAT is an Open Source C++ Toolbox for SMT solving and is capable of *strategic* and *parallel* solving. It is under the care of the *Theory of Hybrid Systems* group at RWTH Aachen University. It makes use of different solving techniques each encapsulated in a module and tackling an SMT instance differently and efficiently. For example, the *simplex method* is appropriate for linear real constraints of the form $\sum_{i=1}^{n} a_i \cdot x_i \leq b_i$ while other SAT-solving modules are appropriate for boolean constraints. Each module is designed to work incrementally and support backtracking. The main advantage of this approach is remembering certain formulas and important properties (for example, reasons for unsatisfiability in form of formula sets) from the previously considered constraints. So even if the constraint set is changed by adding or removing formulas, the solver does not need to consider the whole set again but rather utilizes those properties to make significantly faster decisions.

Although we are not successful in advancing SMT-RAT by the solving technique proposed by Dr. Rahkooy, during our research and discussions another idea came up which we tested and analyzed statistics of. It consists of increasing the efficiency of the subtropical module of SMT-RAT by changing the reduction to an LRA instance slightly.
Currently, the subtropical module constructs a hyperplane formula using the described methods in [FOSV17]. It is capable of finding general solutions most of the times (due to incompleteness) for multiple polynomial inequations. For reference, the subtropical real root finding method failed only in less than 8 percent of several hundreds of benchmarks with over 800.000 monomials in up to 10 variables with a maximum degree of 12 [Stu15]. In order to solve constraints of the form $f \neq 0$ the module splits that constraint into $f < 0 \oplus f > 0$ (XOR). The hypothesis in our discussion was whether the module performs differently if we instead transform such constraints into $yf > 0$ where y is a fresh new variable. The difference being is that for n constraints of this form instead of considering $2n$ constraints, we only check n constraints with n new variables.

4.1 Tests

Notice: The tests were measured on a local machine and do not represent SMT-RAT's performance in a competitive sense and should therefore not be used to compare SMT-RAT to other SMT solvers that are not tested under the same conditions. They are merely shown to compare the performance of different methods within SMT-RAT.

For the analysis we used over 10,000 benchmarks of QF NRA formulas from SMT-LIB, an international facility that thrives to improve research and development of SMT solving and provides a large variety of SMT benchmarks [smtb] [smta]. In order to measure the performance of the subtropical module precisely, the strategy is only composed of the SAT-module followed by the subtropical module. The tests were conducted on a local machine provided with 3 gigabytes and 20 seconds of computation time.

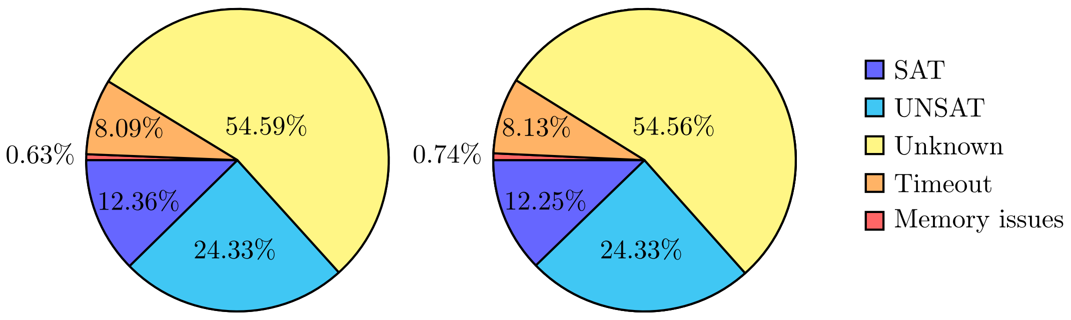

Figure 4.1: Results of the original subtropical method (left) and variation (right). Rounded to four decimals. 10,668 benchmarks used.

The comparison shows that there is no *significant* difference between the computed answers, however the slight contrast between variation and original hints that SMT-RAT performs worse with the variation. This is already reason enough to not introduce the proposed change. The tables 4.1 and 4.2 show the difference in time and memory performance:

Time performance

Subtropical module	Min.	Max.	Avg.	Median	LQ	UQ
Original	12	20,094	1,857.4	17	15	91
Variation	12	20,084	1,859.28	17	14	94

Table 4.1: Computation time in milliseconds (LQ/UQ = lower/upper quartal)

Memory performance

Subtropical module	Min.	Max.	Avg.	Median	LQ	UQ
Original	19,432	3,131,648	161,566.21	20,984	20,124	27,804
Variation	19,372	3,131,664	161,721.67	21,008	20,148	28,416

Table 4.2: Required memory in kilobytes

4.2 Other Implementations

Within this section we want to present some other implementations that are a byproduct of this research.

4.2.1 Transformation to a NRA formula

We programmed a method that realizes the transformation in *Section 3.2.1* which introduces a quantifier to the formula but also transforms the polynomial constraint set Φ into a single equisatisfiable formula ψ. This can be done in two steps:

1. Create a polynomial set F and transform each constraint φ in Φ into an equisatisfiable polynomial equational constraint φ' using the transformation rules 1 to 7 in *Section 3*. Let g be the polynomial that appear in the transformed constraint φ' and y a fresh new variable. Add for each transformed constraint the polynomial $(y \cdot g + 1)$ to F.

2. Finally, construct $\psi = (\forall y (\prod_{f \in F} f) - 1 = 0)$.

Similarly, we used over 10,000 benchmarks to measure the performance of such a transformation. We also used 3 gigabytes of memory but increased the computation time available from 20 seconds to 1 minute. All transformation steps are the same with the only exception that a quantifier is added. We can therefore get at least a rough expectation of how much computation time and memory are required for a similar transformation.

Min.	Max.	Avg.	Median	LQ	UQ
12	60,216	19,169.32	213	28	60,062

Table 4.3: Computation time in milliseconds

Note: From 11,552 benchmarks around 3,314 (which is approx. 29%) resulted in timeouts.

Min.	Max.	Avg.	Median	LQ	UQ
18,924	3,130,748	731,054.78	43,012	21,128	1,774,060

Table 4.4: Required memory in kilobytes

4.2.2 Edge case: Single Polynomial Equation

In the following, we depict some statistics of satisfiability checking, described in this thesis, for the rare case that the polynomial constraint set is a single equation. Equivalently, we provide statistics for the subtropical real root finding method described in [Stu15] [FOSV17]. It is questionable if such a module even fits into the structure for SMT-RAT or SMT solving in general as most SMT instances won't consist of a singular constraint. That is why we compare our method with the strategy *SMTCOMP* of SMT-RAT. The strategy is composed of 22 modules at the time of our research. For the benchmarks, we used 3,000 satisfiable SMT instances with up to 10 monomials, 5 variables, and an expected degree of 15 (max. 40). Both strategies were given 3 gigabytes of memory and 1 minute of computation time.

	SAT	Unknown	Timeout and other	Total time	Avg. time
Default strategy	2,391	0	609	37,137,210	12,379.07
Proposed method	2,974	0	26	2,132,670	710.89

Table 4.5: Comparison of SMT-RAT default strategy and proposed method for single constraints

We can see that at least for the edge case that SMT-RAT with the proposed method for polynomial constraint singletons is significantly performing better in terms of computation time and gathered answers. The results also agree with the results of [Stu15][FOSV17] that the subtropical method is indeed significantly more efficient. There is a total time difference of 583 minutes and approximately 25 seconds (around 10 hours in total). The method therefore at least provides great utility for such edge cases which may not appear in general SMT instances but could be useful for determining satisfiability of singular constraints whose variables do not originate from the considered SMT instance.

An efficient transformation that transforms polynomial constraint sets into a single equisatisfiable quantifier-free polynomial equation while also not restricting the output space to solely non-negative values (or non-positive values) could lead to significant computation time improvements for SMT solving.

Chapter 5

Conclusion

In conclusion, we attempted to advance SMT-RAT and SMT solving in general by the capability of solving QF NRA using the subtropical method without being restricted to disregard conjunctions that include polynomial equational constraints. We introduced the subtropical real root finding method [Stu15] [FOSV17] and the cylindrical algebraic decomposition [Jir95] for univariate polynomial constraint sets. The analysis of the transformation showed that the initial idea regarding the conjunction rule does not yield an equisatisfiable formula which is an absolute necessity. Upon that, we elaborated alternative approaches such as the Gröbner base and gradient equation solving. We analyzed other closely related implementations using statistical measures to check for efficiency improvements within the subtropical satisfiability, to measure the computation time for a transformation that introduces quantifiers, and to see how SMT-RAT with the proposed method compares to another of its strategies in an edge case.
The empirical results of the last tests coincide with the results in [FOSV17] and [Stu15] and shows: If future research ever finds a possibility to realize such a transformation efficiently then the performance for SMT solving of NRA formulas could be greatly improved.

Even though we are not successful to realize this idea, we still hope that this thesis provides new perspectives for the development of SMT solving.

Bibliography

[Abe24] Niels Henrik Abel. *Mémoire sur les équations algébriques, où on demontre l'impossibilité de la résolution de l'équation générale du cinquième dégré.* 1824.

[Abe26] Niels Henrik Abel. Démonstration de l'impossibilité de la résolution algébrique des équations générales qui passent le quatrieme degré. *Journal für die reine und angewandte Mathematik*, 1:65–96, 1826.

[Ayo80] Raymond G Ayoub. Paolo ruffini's contributions to the quintic. *Archive for history of exact sciences*, pages 253–277, 1980.

[BK10] Bruno Buchberger and Manuel Kauers. Gröbner basis. *Scholarpedia*, 5(10):7763, 2010.

[Col75] George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata theory and formal languages*, pages 134–183. Springer, 1975.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of Computing*, pages 151–158, 1971.

[Fit12] Melvin Fitting. *First-order logic and automated theorem proving.* Springer Science & Business Media, 2012.

[FOSV17] Pascal Fontaine, Mizuhito Ogawa, Thomas Sturm, and Xuan Tung Vu. Subtropical satisfiability. In *International Symposium on Frontiers of Combining Systems*, pages 189–206. Springer, 2017.

[Grä11] Erich Grädel. Mathematische logik. *Vorlesungsmanuskript, RWTH Aachen*, 2011.

[Jir95] Mats Jirstrand. *Cylindrical algebraic decomposition - an introduction.* Linköping University, 1995.

[LT93] Nancy G Leveson and Clark S Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[maea] Maeslantkering barrier - parts.

[maeb] Rekenkracht is Macht.

[Neu18] Tom Neuhäuser. *Quantifier Elimination by Cylindrical Algebraic Decomposition.* Bachelor's thesis. RWTH Aachen University. 2018.

[oM] Encyclopedia of Mathematics. Buchberger algorithm. "http://encyclopediaofmath.org/index.php?title=Buchberger_algorithm&oldid=46215".

[Rag] K. N. Raghavan. Sturm's method for the number of real roots of a real polynomial. "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjR5dfVl-f2AhUOSvEDHQjeDmkQFnoECAUQAQ&url=https%3A%2F%2Fwww.imsc.res.in%2F~knr%2Fpast%2Fsturm%2Fformal_notes.pdf&usg=AOvVaw0hC_onApq2cdAuf4mn-kJH".

[RAUa] Algorithmen und Komplexität RWTH Aachen University, Lehrstuhl für Informatik 1. Vorlesung Effiziente Algorithmen.

[RAUb] Theory of Hybrid Systems RWTH Aachen University. Vorlesung satisfiability checking.

[Ruf13] Paolo Ruffini. *Riflessioni intorno alla soluzione delle equazioni algebraiche generali opuscolo del cav. dott. Paolo Ruffini...* presso la Societa Tipografica, 1813.

[smta] QF NRA benchmarks. "https://clc-gitlab.cs.uiowa.edu:2443/SMT-LIB-benchmarks/QF_NRA".

[smtb] The satisfiability module theories library. "https://smtlib.cs.uiowa.edu/".

[Sta] CBS Statline. South holland population data table. "https://opendata.cbs.nl/statline/#/CBS/nl/dataset/37230ned/table".

[Stu09] Par C Sturm. Mémoire sur la résolution des équations numériques. In *Collected Works of Charles François Sturm*, pages 345–390. Springer, 2009.

[Stu15] Thomas Sturm. Subtropical real root finding. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 347–354, 2015.

[sul15] sulky. Gemeinsame Nullstellen von Polynom und Ableitung. "https://www.matheplanet.com/default3.html?call=viewtopic.php?topic=212242", 2015.

[Tar98] Alfred Tarski. A decision method for elementary algebra and geometry. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 24–84. Springer, 1998.